BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images may include (but are not limited to):

- BLACK BORDERS
- TEXT CUT OFF AT TOP, BOTTOM OR SIDES
- FADED TEXT
- ILLEGIBLE TEXT
- SKEWED/SLANTED IMAGES
- COLORED PHOTOS
- BLACK OR VERY BLACK AND WHITE DARK PHOTOS
- GRAY SCALE DOCUMENTS

IMAGES ARE BEST AVAILABLE COPY.

As rescanning documents *will not* correct images, please do not report the images to the Image Problem Mailbox.

INTERNATIONAL TELECOMMUNICATION UNION

ITU-T G.729

(03/96)

TELECOMMUNICATION
STANDARDIZATION SECTOR
OF ITU

GENERAL ASPECTS OF DIGITAL TRANSMISSION SYSTEMS

CODING OF SPEECH AT 8 kbit/s USING CONJUGATE-STRUCTURE ALGEBRAIC-CODE-EXCITED LINEAR-PREDICTION (CS-ACELP)

ITU-T Recommendation G.729

(Previously "CCITT Recommendation")

FOREWORD

The ITU-T (Telecommunication Standardization Sector) is a permanent organ of the International Telecommunication Union (ITU). The ITU-T is responsible for studying technical, operating and tariff questions and issuing Recommendations on them with a view to standardizing telecommunications on a worldwide basis.

The World Telecommunication Standardization Conference (WTSC), which meets every four years, establishes the topics for study by the ITU-T Study Groups which, in their turn, produce Recommendations on these topics.

The approval of Recommendations by the Members of the ITU-T is covered by the procedure laid down in WTSC Resolution No. 1 (Helsinki, March 1-12, 1993).

ITU-T Recommendation G.729 was prepared by ITU-T Study Group 15 (1993-1996) and was approved under the WTSC Resolution No. 1 procedure on the 19th of March 1996.

NOTE

In this Recommendation, the expression "Administration" is used for conciseness to indicate both a telecommunication administration and a recognized operating agency.

CONTENTS

CODING OF SPEECH AT 8 kbit/s USING CONJUGATE-STRUCTURE ALGEBRAIC-CODE-EXCITED LINEAR-PREDICTION (CS-ACELP)

(Geneva, 1996)

1 Introduction

This Recommendation contains the description of an algorithm for the coding of speech signals at 8 kbit/s using Conjugate-Structure Algebraic-Code-Excited Linear-Prediction (CS-ACELP).

This coder is designed to operate with a digital signal obtained by first performing telephone bandwidth filtering (Recommendation G.712) of the analogue input signal, then sampling it at 8000 Hz, followed by conversion to 16-bit linear PCM for the input to the encoder. The output of the decoder should be converted back to an analogue signal by similar means. Other input/output characteristics, such as those specified by Recommendation G.711 for 64 kbit/s PCM data, should be converted to 16-bit linear PCM before encoding, or from 16-bit linear PCM to the appropriate format after decoding. The bitstream from the encoder to the decoder is defined within this Recommendation.

This Recommendation is organized as follows: Clause 2 gives a general outline of the CS-ACELP algorithm. In clauses 3 and 4, the CS-ACELP encoder and decoder principles are discussed, respectively. Clause 5 describes the software that defines this coder in 16 bit fixed-point arithmetic.

2 General description of the coder

The CS-ACELP coder is based on the Code-Excited Linear-Prediction (CELP) coding model. The coder operates on speech frames of 10 ms corresponding to 80 samples at a sampling rate of 8000 samples per second. For every 10 ms frame, the speech signal is analysed to extract the parameters of the CELP model (linear-prediction filter coefficients, adaptive and fixed-codebook indices and gains). These parameters are encoded and transmitted. The bit allocation of the coder parameters is shown in Table 1. At the decoder, these parameters are used to retrieve the excitation and synthesis filter parameters. The speech is reconstructed by filtering this excitation through the short-term synthesis filter, as is shown in Figure 1. The short-term synthesis filter is based on a 10th order Linear Prediction (LP) filter. The long-term, or pitch synthesis filter is implemented using the so-called adaptive-codebook approach. After computing the reconstructed speech, it is further enhanced by a postfilter.

TABLE 1/G.729

Bit allocation of the 8 kbit/s CS-ACELP algorithm (10 ms frame)

Parameter	Codeword	Subframe 1	Subframe 2	Total per frame
Line spectrum pairs	*L0*, *L1*, *L2*, *L3*			18
Adaptive-codebook delay	*P1*, *P2*	8	5	13
Pitch-delay parity	*P0*	1		1
Fixed-codebook index	*C1*, *C2*	13	13	26
Fixed-codebook sign	*S1*, *S2*	4	4	8
Codebook gains (stage 1)	*GA1*, *GA2*	3	3	6
Codebook gains (stage 2)	*GB1*, *GB2*	4	4	8
Total				80

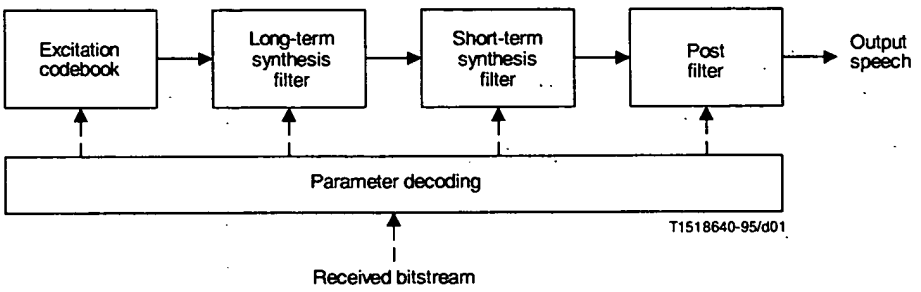

FIGURE 1/G.729

Block diagram of conceptual CELP synthesis model

2.1 Encoder

The encoding principle is shown in Figure 2. The input signal is high-pass filtered and scaled in the pre-processing block. The pre-processed signal serves as the input signal for all subsequent analysis. LP analysis is done once per 10 ms frame to compute the LP filter coefficients. These coefficients are converted to Line Spectrum Pairs (LSP) and quantized using predictive two-stage Vector Quantization (VQ) with 18 bits. The excitation signal is chosen by using an analysis-by-synthesis search procedure in which the error between the original and reconstructed speech is minimized according to a perceptually weighted distortion measure. This is done by filtering the error signal with a perceptual weighting filter, whose coefficients are derived from the unquantized LP filter. The amount of perceptual weighting is made adaptive to improve the performance for input signals with a flat frequency-response.

The excitation parameters (fixed and adaptive-codebook parameters) are determined per subframe of 5 ms (40 samples) each. The quantized and unquantized LP filter coefficients are used for the second subframe, while in the first subframe interpolated LP filter coefficients are used (both quantized and unquantized). An open-loop pitch delay is estimated once per 10 ms frame based on the perceptually weighted speech signal. Then the following operations are repeated for each subframe. The target signal $x(n)$ is computed by filtering the LP residual through the weighted synthesis filter $W(z)/\hat{A}(z)$. The initial states of these filters are updated by filtering the error between LP residual and excitation. This is equivalent to the common approach of subtracting the zero-input response of the weighted synthesis filter from the weighted speech signal. The impulse response $h(n)$ of the weighted synthesis filter is computed. Closed-loop pitch analysis is then done (to find the adaptive-codebook delay and gain), using the target $x(n)$ and impulse response $h(n)$, by searching around the value of the open-loop pitch delay. A fractional pitch delay with 1/3 resolution is used. The pitch delay is encoded with 8 bits in the first subframe and differentially encoded with 5 bits in the second subframe. The target signal $x(n)$ is updated by subtracting the (filtered) adaptive-codebook contribution, and this new target, $x'(n)$, is used in the fixed-codebook search to find the optimum excitation. An algebraic codebook with 17 bits is used for the fixed-codebook excitation. The gains of the adaptive and fixed-codebook contributions are vector quantized with 7 bits, (with MA prediction applied to the fixed-codebook gain). Finally, the filter memories are updated using the determined excitation signal.

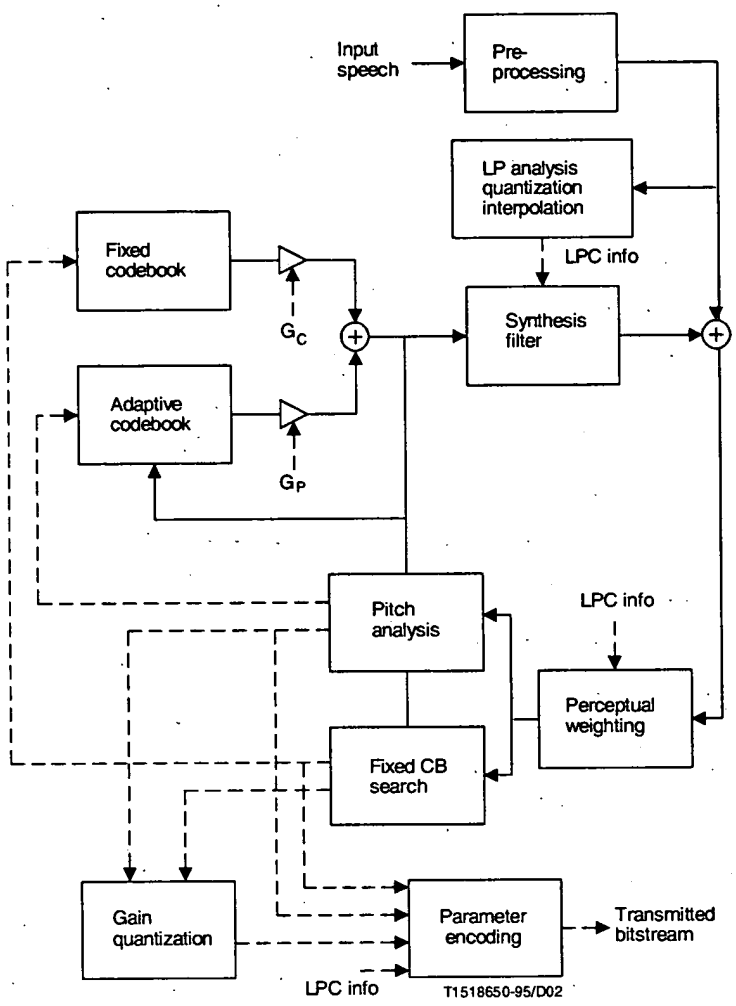

FIGURE 2/G.729

Encoding principle of the CS-ACELP encoder

2.2 Decoder

The decoder principle is shown in Figure 3. First, the parameter's indices are extracted from the received bitstream. These indices are decoded to obtain the coder parameters corresponding to a 10 ms speech frame. These parameters are the LSP coefficients, the two fractional pitch delays, the two fixed-codebook vectors, and the two sets of adaptive and fixed-codebook gains. The LSP coefficients are interpolated and converted to LP filter coefficients for each subframe. Then, for each 5 ms subframe the following steps are done:

- the excitation is constructed by adding the adaptive and fixed-codebook vectors scaled by their respective gains;
- the speech is reconstructed by filtering the excitation through the LP synthesis filter;
- the reconstructed speech signal is passed through a post-processing stage, which includes an adaptive postfilter based on the long-term and short-term synthesis filters, followed by a high-pass filter and scaling operation.

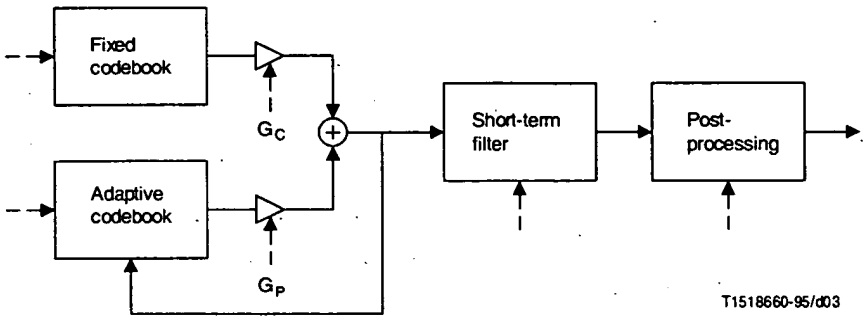

FIGURE 3/G.729

Principle of the CS-ACELP decoder

2.3 Delay

This coder encodes speech and other audio signals with 10 ms frames. In addition, there is a look-ahead of 5 ms, resulting in a total algorithmic delay of 15 ms. All additional delays in a practical implementation of this coder are due to:

- processing time needed for encoding and decoding operations;
- transmission time on the communication link;
- multiplexing delay when combining audio data with other data.

2.4 Speech coder description

The description of the speech coding algorithm of this Recommendation is made in terms of bit-exact, fixed-point mathematical operations. The ANSI C code indicated in clause 5, which constitutes an integral part of this Recommendation, reflects this bit-exact, fixed-point descriptive approach. The mathematical descriptions of the encoder (clause 3), and decoder (clause 4), can be implemented in several other fashions, possibly leading to a codec implementation not complying with this Recommendation. Therefore, the algorithm description of the ANSI C code of clause 5 shall take precedence over the mathematical descriptions of clauses 3 and 4 whenever discrepancies are found. A non-exhaustive set of test signals, which can be used with ANSI C code, are available from the ITU.

2.5 Notational conventions

Throughout this Recommendation, it is tried to maintain the following notational conventions:

- Codebooks are denoted by caligraphic characters (e.g. $\mathcal{C}$).
- Time signals are denoted by their symbol and a sample index between parenthesis [e.g. $s(n)$]. The symbol n is used as sample index.
- Superscript indices between parenthesis (e.g. $g^{(m)}$ are used to indicate time-dependency of variables. The variable m refers, depending on the context, to either a frame or subframe index, and the variable n to a sample index.
- Recursion indices are identified by a superscript between square brackets (e.g. $E^{[k]}$).
- Subscripts indices identify a particular element in a coefficient array.
- The symbol ^ identifies a quantized version of a parameter (e.g. $\hat{g}_c$).
- Parameter ranges are given between square brackets, and include the boundaries (e.g. [0.6, 0.9]).

- The function *log* denotes a logarithm with base 10.
- The function *int* denotes truncation to its integer value.
- The decimal floating-point numbers used are rounded versions of the values used in the 16 bit fixed-point ANSI C implementation.

Table 2 lists the most relevant symbols used throughout this Recommendation. A glossary of the most relevant signals is given in Table 3. Table 4 summarizes relevant variables and their dimension. Constant parameters are listed in Table 5. The acronyms used in this Recommendation are summarized in Table 6.

TABLE 2/G.729

Glossary of most relevant symbols

Name	Reference	Description
$1/\hat{A}(z)$	Equation (2)	LP synthesis filter
$H_{h1}(z)$	Equation (1)	Input high-pass filter
$H_p(z)$	Equation (78)	Long-term postfilter
$H_f(z)$	Equation (84)	Short-term postfilter
$H_t(z)$	Equation (86)	Tilt-compensation filter
$H_{h2}(z)$	Equation (91)	Output high-pass filter
$P(z)$	Equation (46)	Pre-filter for fixed codebook
$W(z)$	Equation (27)	Weighting filter

TABLE 3/G.729

Glossary of most relevant signals

Name	Reference	Description
$c(n)$	3.8	Fixed-codebook contribution
$d(n)$	3.8.1	Correlation between target signal and $h(n)$
$ew(n)$	3.10	Error signal
$h(n)$	3.5	Impulse response of weighting and synthesis filters
$r(n)$	3.6	Residual signal
$s(n)$	3.1	Pre-processed speech signal
$\hat{s}(n)$	4.1.6	Reconstructed speech signal
$s'(n)$	3.2.1	Windowed speech signal
$sf(n)$	4.2	Postfiltered output
$sf'(n)$	4.2	Gain-scaled postfiltered output
$sw(n)$	3.6	Weighted speech signal
$x(n)$	3.6	Target signal
$x'(n)$	3.8.1	Second target signal
$u(n)$	3.10	Excitation to LP synthesis filter
$v(n)$	3.7.1	Adaptive-codebook contribution
$y(n)$	3.7.3	Convolution $v(n) * h(n)$
$z(n)$	3.9	Convolution $c(n) * h(n)$

TABLE 4/G.729

Glossary of most relevant variables

Name	Size	Description
g_p	1	Adaptive-codebook gain
g_c	1	Fixed-codebook gain
g_l	1	Gain term for long-term postfilter
g_f	1	Gain term for short-term postfilter
g_t	1	Gain term for tilt postfilter
G	1	Gain for gain normalization
T_{op}	1	Open-loop pitch delay
a_i	11	LP coefficients ($a_0 = 1.0$)
k_i	10	Reflection coefficients
k'_1	1	Reflection coefficient for tilt postfilter
o_i	2	LAR coefficients
ω_i	10	LSF normalized frequencies
$\hat{p}_{i,j}$	40	MA predictor for LSF quantization
q_i	10	LSP coefficients
$r(k)$	11	Auto-correlation coefficients
$r'(k)$	11	Modified auto-correlation coefficients
w_i	10	LSP weighting coefficients
$\hat{l}_i$	10	LSP quantizer output

TABLE 5/G.729

Glossary of most relevant constants

Name	Value	Description
f_s	8000	Sampling frequency
f_0	60	Bandwidth expansion
γ_1	0.94/0.98	Weight factor perceptual weighting filter
γ_2	0.60/[0.4 – 0.7]	Weight factor perceptual weighting filter
γ_n	0.55	Weight factor postfilter
γ_d	0.70	Weight factor postfilter
γ_p	0.50	Weight factor pitch postfilter
γ_t	0.90/0.2	Weight factor tilt postfilter
$\mathcal{C}$	Table 7	Fixed (algebraic) codebook
L0	3.2.4	Moving-average predictor codebook
L1	3.2.4	First stage LSP codebook
L2	3.2.4	Second stage LSP codebook (low part)
L3	3.2.4	Second stage LSP codebook (high part)
$\mathcal{GA}$	3.9	Gain codebook (first stage)
$\mathcal{GB}$	3.9	Gain codebook (second stage)
w_{lag}	Equation (6)	Correlation lag window
w_{lp}	Equation (3)	LP analysis window

TABLE 6/G.729

Glossary of acronyms

Acronym	Description
CELP	Code-Excited Linear-Prediction
CS-ACELP	Conjugate-Structure Algebraic-CELP
MA	Moving Average
MSB	Most Significant Bit
MSE	Mean-Squared Error
LAR	Log Area Ratio
LP	Linear Prediction
LSP	Line Spectral Pair
LSF	Line Spectral Frequency
VQ	Vector quantization

3 Functional description of the encoder

In this clause the different functions of the encoder represented in the blocks of Figure 2 are described. A detailed signal flow is shown in Figure 4.

3.1 Pre-processing

As stated in clause 2, the input to the speech encoder is assumed to be a 16 bit PCM signal. Two pre-processing functions are applied before the encoding process:

1) signal scaling; and
2) high-pass filtering.

The scaling consists of dividing the input by a factor 2 to reduce the possibility of overflows in the fixed-point implementation. The high-pass filter serves as a precaution against undesired low-frequency components. A second order pole/zero filter with a cut-off frequency of 140 Hz is used. Both the scaling and high-pass filtering are combined by dividing the coefficients at the numerator of this filter by 2. The resulting filter is given by:

$$H_{h1}(z) = \frac{0.46363718 - 0.92724705z^{-1} + 0.46363718z^{-2}}{1 - 1.9059465z^{-1} + 0.9114024z^{-2}} \quad (1)$$

The input signal filtered through $H_{h1}(z)$ is referred to as $s(n)$, and will be used in all subsequent coder operations.

3.2 Linear prediction analysis and quantization

The short-term analysis and synthesis filters are based on 10th order Linear Prediction (LP) filters.

The LP synthesis filter is defined as:

$$\frac{1}{\hat{A}(z)} = \frac{1}{1 + \sum_{i=1}^{10} \hat{a}_i z^{-i}} \quad (2)$$

where $\hat{a}_i$, $i = 1,...,10$, are the (quantized) Linear Prediction (LP) coefficients. Short-term prediction, or linear prediction analysis is performed once per speech frame using the autocorrelation method with a 30 ms asymmetric window. Every 80 samples (10 ms), the autocorrelation coefficients of windowed speech are computed and converted to the LP coefficients using the Levinson algorithm. Then the LP coefficients are transformed to the LSP domain for quantization and interpolation purposes. The interpolated quantized and unquantized filters are converted back to the LP filter coefficients (to construct the synthesis and weighting filters for each subframe).

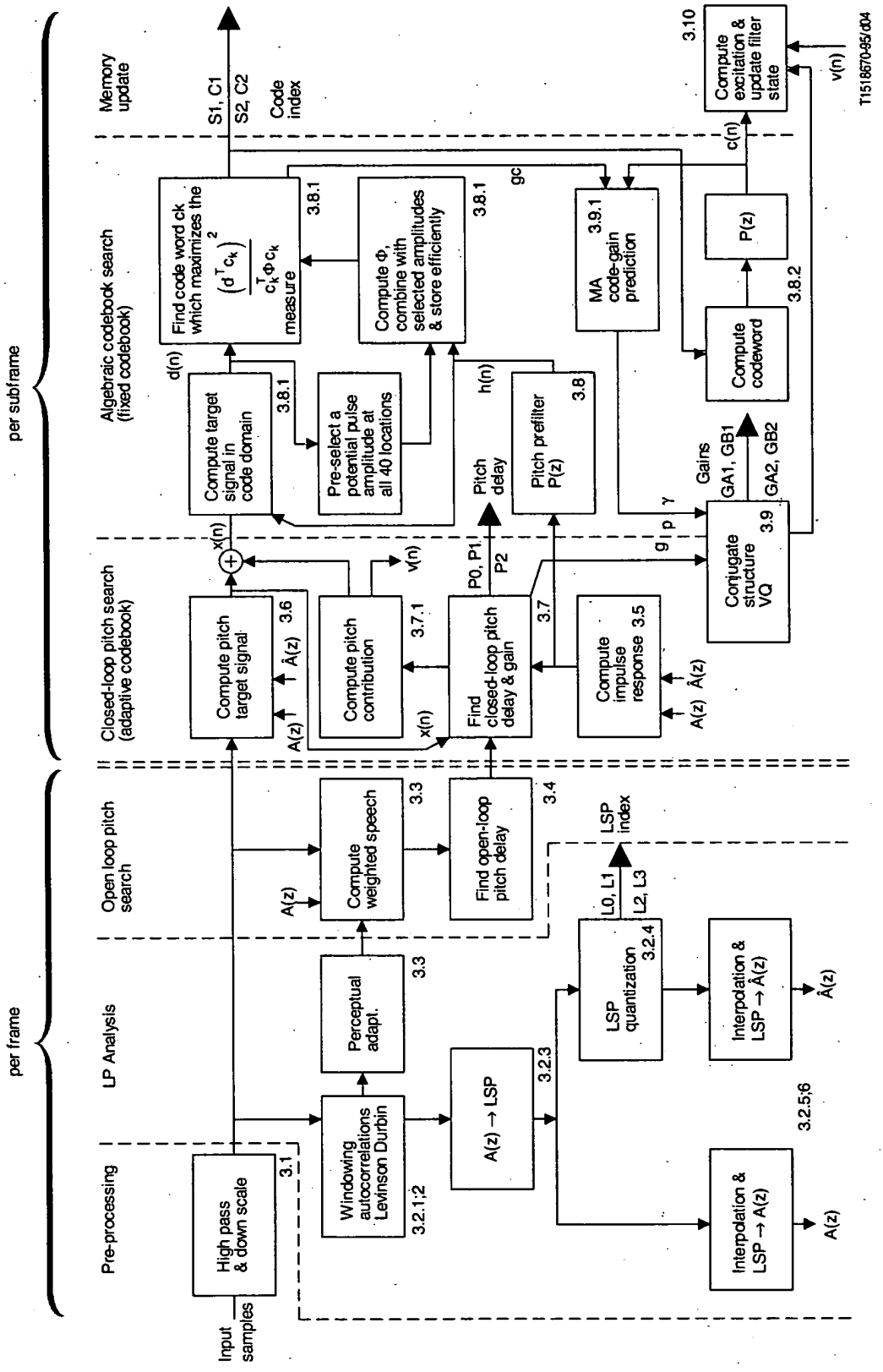

FIGURE 4/G.729

Signal flow at the CS-ACELP encoder

3.2.1 Windowing and autocorrelation computation

The LP analysis window consists of two parts: the first part is half a Hamming window and the second part is a quarter of a cosine function cycle. The window is given by:

$$w_{lp}(n) = \begin{cases} 0.54 - 0.46\cos\left(\dfrac{2\pi n}{399}\right) & n = 0,...,199 \\ \cos\left(\dfrac{2\pi\,(n - 200)}{159}\right) & n = 200,...,239 \end{cases} \tag{3}$$

There is a 5 ms lookahead in the LP analysis which means that 40 samples are needed from the future speech frame. This translates into an extra algorithmic delay of 5 ms at the encoder stage. The LP analysis window applies to 120 samples from past speech frames, 80 samples from the present speech frame, and 40 samples from the future frame. The windowing procedure is illustrated in Figure 5.

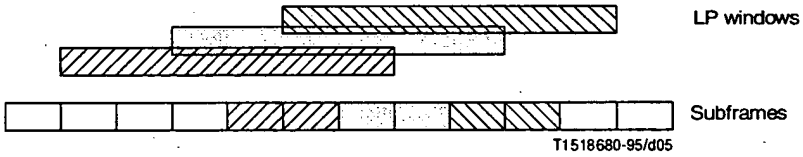

FIGURE 5/G.729

Windowing procedure in LP analysis

The different shading patterns identify corresponding excitation and LP analysis windows.

The windowed speech:

$$s'(n) = w_{lp}(n)\, s(n) \qquad n = 0,...,239 \tag{4}$$

is used to compute the autocorrelation coefficients:

$$r(k) = \sum_{n=k}^{239} s'(n)\, s'(n - k) \qquad k = 0,...,10 \tag{5}$$

To avoid arithmetic problems for low-level input signals the value of $r(0)$ has a lower boundary of $r(0) = 1.0$. A 60 Hz bandwidth expansion is applied, by multiplying the autocorrelation coefficients with:

$$w_{lag}(k) = \exp\left[-\frac{1}{2}\left(\frac{2\pi f_0 k}{f_s}\right)^2\right] \qquad k = 1,...,10 \tag{6}$$

where $f_0 = 60$ Hz is the bandwidth expansion and $f_s = 8000$ Hz is the sampling frequency. Furthermore, $r(0)$ is multiplied by a white-noise correction factor 1.0001, which is equivalent to adding a noise floor at –40 dB. The modified autocorrelation coefficients are given by:

$$\begin{aligned} r'(0) &= 1.0001\, r(0) \\ r'(k) &= w_{lag}(k)\, r(k) \qquad k = 1,...,10 \end{aligned} \tag{7}$$

3.2.2 Levinson-Durbin algorithm

The modified autocorrelation coefficients $r'(k)$ are used to obtain the LP filter coefficients, a_i, $i = 1,...,10$, by solving the set of equations:

$$\sum_{i=1}^{10} a_i r'(|i - k|) = -r'(k) \qquad k = 1,...,10 \tag{8}$$

The set of equations in (8) is solved using the Levinson-Durbin algorithm. This algorithm uses the following recursion:

$E^{[0]} = r'(0)$

for $i = 1$ *to* 10

$a_0^{[i-1]} = 1$

$k_i = -\left[\sum_{j=0}^{i-1} a_j^{[i-1]} r'(i-j)\right] / E^{[i-1]}$

$a_i^{[i]} = k_i$

for $j = 1$ *to* $i - 1$

$a_j^{[i]} = a_j^{[i-1]} + k_i a_{i-j}^{[i-1]}$

end

$E^{[i]} = \left(1 - k_i^2\right) E^{[i-1]}$

end

The final solution is given as $a_j = a_j^{[10]}$, $j = 0, ...,10$, with $a_0 = 1.0$.

3.2.3 LP to LSP conversion

The LP filter coefficients a_i, $i = 0,...10$ are converted to Line Spectral Pair (LSP) coefficients for quantization and interpolation purposes. For a 10th order LP filter, the LSP coefficients are defined as the roots of the sum and difference polynomials:

$$F_1'(z) = A(z) + z^{-11}A(z^{-1}) \tag{9}$$

and:

$$F_2'(z) = A(z) - z^{-11}A(z^{-1}) \tag{10}$$

respectively. The polynomial $F_1'(z)$ is symmetric, and $F_2'(z)$ is antisymmetric. It can be proven that all roots of these polynomials are on the unit circle and they alternate each other. $F_1'(z)$ has a root $z = -1$ ($\omega = \pi$) and $F_2'(z)$ has a root $z = 1$ ($w = 0$). These two roots are eliminated by defining the new polynomials:

$$F_1(z) = F_1'(z) / (1 + z^{-1}) \tag{11}$$

and:

$$F_2(z) = F_2'(z) / (1 - z^{-1}) \tag{12}$$

Each polynomial has five conjugate roots on the unit circle ($e^{\pm j\omega_i}$), and they can be written as:

$$F_1(z) = \prod_{i=1,3,...,9} (1 - 2q_i z^{-1} + z^{-2}) \tag{13}$$

and:

$$F_2(z) = \prod_{i=2,4,...,10} (1 - 2q_i z^{-1} + z^{-2}) \tag{14}$$

where $q_i = \cos(\omega_i)$. The coefficients ω_i are the Line Spectral Frequencies (LSF) and they satisfy the ordering property $0 < \omega_i < \omega_2 < ... < \omega_{10} < \pi$. The coefficients q_i are referred to as the LSP coefficients in the cosine domain.

Since both polynomials $F_1(z)$ and $F_2(z)$ are symmetric only the first five coefficients of each polynominal need to be computed. The coefficients of these polynomials are found by the recursive relations:

$$\begin{aligned} f_1(i + 1) &= a_{i+1} + a_{10-i} - f_1(i) \qquad i = 0,...,4 \\ f_2(i + 1) &= a_{i+1} - a_{10-i} + f_2(i) \qquad i = 0,...,4 \end{aligned} \tag{15}$$

where $f_1(0) = f_2(0) = 1.0$. The LSP coefficients are found by evaluating the polynomials $F_1(z)$ and $F_2(z)$ at 60 points equally spaced between 0 and π and checking for sign changes. A sign change signifies the existence of a root and the sign change interval is then divided four times to allow better tracking of the root. The Chebyshev polynomials are used to evaluate $F_1(z)$ and $F_2(z)$. In this method the roots are found directly in the cosine domain. The polynomials $F_1(z)$ or $F_2(z)$, evaluated at $z = e^{j\omega}$, can be written as:

$$F(\omega) = 2e^{-j5\omega} C(x) \tag{16}$$

with:

$$C(x) = T_5(x) + f(1)T_4(x) + f(2)T_3(x) + f(3)T_2(x) + f(4)T_1(x) + f(5)/2 \tag{17}$$

where $T_m(x) = \cos(m\omega)$ is the mth order Chebyshev polynomial, and $f(i)$, $i = 1,...,5$, are the coefficients of either $F_1(z)$ or $F_2(z)$, computed using Equation (15). The polynomial $C(x)$ is evaluated at a certain value of $x = \cos(\omega)$ using the recursive relation:

for $k = 4$ *down to* 1

$$b_k = 2xb_{k+1} - b_{k+2} + f(5 - k)$$

end

$$C(x) = xb_1 - b_2 + f(5)/2$$

with initial values $b_5 = 1$ and $b_6 = 0$.

3.2.4 Quantization of the LSP coefficients

The LSP coefficients q_i are quantized using the LSF representation ω_i in the normalized frequency domain [0, π]; that is:

$$\omega_i = \arccos(q_i) \qquad i = 1,...,10 \tag{18}$$

A switched 4th order MA prediction is used to predict the LSF coefficients of the current frame. The difference between the computed and predicted coefficients is quantized using a two-stage vector quantizer. The first stage is a 10-dimensional VQ using codebook L1 with 128 entries (7 bits). The second stage is a 10 bit VQ which has been implemented as a split VQ using two 5-dimensional codebooks, L2 and L3 containing 32 entries (5 bits) each.

To explain the quantization process, it is convenient to first describe the decoding process. Each coefficient is obtained from the sum of two codebooks:

$$\hat{l}_i = \begin{cases} \mathrm{L1}_i(L1) + \mathrm{L2}_i(L2) & i = 1,...,5 \\ \mathrm{L1}_i(L1) + \mathrm{L3}_{i-5}(L3) & i = 6,...,10 \end{cases} \tag{19}$$

where $L1$, $L2$ and $L3$ are the codebook indices. To avoid sharp resonances in the quantized LP synthesis filter, the coefficients $\hat{l}_i$ are arranged such that adjacent coefficients have a minimum distance of J. The rearrangement routine is shown below:

for $i = 2,...,10$
  if $(\hat{l}_{i-1} > \hat{l}_i - J)$
    $\hat{l}_{i-1} = (\hat{l}_i + \hat{l}_{i-1} - J)/2$
    $\hat{l}_i = (\hat{l}_i + \hat{l}_{i-1} + J)/2$
  end
end

This rearrangement process is done twice. First with a value of $J = 0.0012$, then with a value of $J = 0.0006$. After this rearrangement process, the quantized LSF coefficients $\hat{\omega}_i^{(m)}$ for the current frame m, are obtained from the weighted sum of previous quantizer outputs $\hat{l}_i^{(m-k)}$, and the current quantizer output $\hat{l}_i^{(m)}$:

$$\hat{\omega}_i^{(m)} = \left(1 - \sum_{k=1}^{4} \hat{p}_{i,k}\right) \hat{l}_i^{(m)} + \sum_{k=1}^{4} \hat{p}_{i,k}\, \hat{l}_i^{(m-k)} \qquad i = 1,...,10 \tag{20}$$

where $\hat{p}_{i,k}$ are the coefficients of the switched MA predictor. Which MA predictor to use is defined by a separate bit $L0$. At start-up the initial values of $\hat{l}_i^{(k)}$ are given by $\hat{l}_i = i\pi/11$ for all $k < 0$.

After computing $\hat{\omega}_i$, the corresponding filter is checked for stability. This is done as follows:

1) order the coefficient $\hat{\omega}_i$ in increasing value;

2) if $\hat{\omega}_i < 0{,}005$ then $\hat{\omega}_i = 0.005$;

3) if $\hat{\omega}_{i+1} - \hat{\omega}_i - 0.0391$ then $\hat{\omega}_{i+1} = \hat{\omega}_i + 0.0391$, $\quad i = 1,...,9$;

4) if $\hat{\omega}_{10} > 3.135$ then $\hat{\omega}_{10} = 3.135$.

The procedure for encoding the LSF parameters can be outlined as follows. For each of the two MA predictors the best approximation to the current LSF coefficients has to be found. The best approximation is defined as the one that minimizes the weighted mean-squared error:

$$E_{lsf} = \sum_{i=1}^{10} w_i(\omega_i - \hat{\omega}_i)^2 \tag{21}$$

The weights w_i are made adaptive as a function of the unquantized LSF coefficients,

$$w_i = \begin{cases} 1.0 & \text{if } \omega_2 - 0.04\pi - 1 > 0 \\ 10\,(\omega_2 - 0.04\pi - 1)^2 + 1 & \text{otherwise} \end{cases}$$

$$w_i \; 2 \le i \le 9 = \begin{cases} 1.0 & \text{if } \omega_{i+1} - \omega_{i-1} - 1 > 0 \\ 10\,(\omega_{i+1} - \omega_{i-1} - 1)^2 + 1 & \text{otherwise} \end{cases} \tag{22}$$

$$w_{10} = \begin{cases} 1.0 & \text{if } -\omega_9 + 0.92\pi - 1 > 0 \\ 10\,(-\omega_9 + 0.92\pi - 1)^2 + 1 & \text{otherwise} \end{cases}$$

In addition, the weights w_5 and w_6 are multiplied by 1.2 each.

The vector to be quantized for the current frame m is obtained from

$$l_i = \left[\omega_i^{(m)} - \sum_{k=1}^{4} \hat{p}_{i,k}\, \hat{l}_i^{(m-k)} \right] / \left(1 - \sum_{k=1}^{4} \hat{p}_{i,k} \right) \qquad i = 1,...,10 \tag{23}$$

The first codebook L1 is searched and the entry $L1$ that minimizes the (unweighted) mean-squared error is selected. This is followed by a search of the second codebook L2, which defines the lower part of the second stage. For each possible candidate, the partial vector $\hat{\omega}_i$, $i = 1,...,5$, is reconstructed using Equation (20), and rearranged to guarantee a minimum distance of 0.0012. The weighted MSE of Equation (21) is computed, and the vector $L2$ which results in the lowest error is selected. Using the selected first stage vector $L1$ and the lower part of the second stage $L2$, the higher part of the second stage is searched from codebook L3. Again the rearrangement procedure is used to guarantee a minimum distance of 0.0012. The vector $L3$ that minimizes the weighted MSE is selected. The resulting vector $\hat{l}_i$, $i = 1,...,10$ is rearranged to guarantee a minimum distance of 0.0006. This process is done for each of the two MA predictors defined by L0, and the MA predictor $L0$ that produces the lowest weighted MSE is selected. As was explained at the beginning of this clause, the resulting vector $\hat{l}_i$ is rearranged twice and a stability check is applied to produce the quantized LSF coefficients $\hat{\omega}_i$.

3.2.5 Interpolation of the LSP coefficients

The quantized (and unquantized) LP coefficients are used for the second subframe. For the first subframe, the quantized (and unquantized) LP coefficients are obtained by linear interpolation of the corresponding parameters in the adjacent subframes. The interpolation is done on the LSP coefficients in the cosine domain. Let $q_i^{(current)}$ be the LSP coefficients computed for the current 10 ms frame, and $q_i^{(previous)}$ the LSP coefficients computed in the previous 10 ms frame. The (unquantized) interpolated LSP coefficients in each of the two subframes are given by:

$$\begin{aligned} &\textit{Subframe } 1\text{:} \quad q_i^{(1)} = 0.5q_i^{(previous)} + 0.5q_i^{(current)} && i = 1,...,10 \\ &\textit{Subframe } 2\text{:} \quad q_i^{(2)} = q_i^{(current)} && i = 1,...,10 \end{aligned} \tag{24}$$

The same interpolation procedure is used for the interpolation of the quantized LSP coefficients by substituting q_i by $\hat{q}_i$ in Equation (24).

3.2.6 LSP to LP conversion

Once the LSP coefficients are quantized and interpolated, they are converted back to the LP coefficients a_i. This conversion is done as follows. The coefficients of $F_1(z)$ and $F_2(z)$ are found by expanding Equations (13) and (14) knowing the quantized and interpolated LSP coefficients. The coefficients $f_1(i)$, $i = 1,...,5$, are computed from q_i using the recursive relation:

for $i = 1$ *to* 5

$$f_1(i) = -2q_{2i-1}f_1(i-1) + 2f_1(i-2)$$

for $j = i - 1$ *down to* 1

$$f_1^{[i]}(j) = f_1^{[i-1]}(j) - 2q_{2i-1}f_1^{[i-1]}(j-1) + f_1^{[i-1]}(j-2)$$

end

end

with initial values $f_1(0) = 1$ and $f_1(-1) = 0$. The coefficients $f_2(i)$ are computed similarly by replacing q_{2i-1} by q_{2i}.

Once the coefficients $f_1(i)$ and $f_2(i)$ are found, $F_1(z)$ and $F_2(z)$ are multiplied by $1 + z^{-1}$ and $1 - z^{-1}$, respectively, to obtain $F_1'(z)$ and $F_2'(z)$; that is:

$$\begin{aligned} f_1'(i) &= f_1(i) + f_1(i-1) \qquad i = 1,...,5 \\ f_2'(i) &= f_2(i) - f_2(i-1) \qquad i = 1,...,5 \end{aligned} \tag{25}$$

Finally the LP coefficients are computed from $f_1'(i)$ and $f_2'(i)$ by:

$$a_i = \begin{cases} 0.5f_1'(i) + 0.5f_2'(i) & i = 1,...,5 \\ 0.5f_1'(11-i) - 0.5f_2'(11-i) & i = 6,...,10 \end{cases} \tag{26}$$

This is directly derived from the relation $A(z) = (F_1'(z) + F_2'(z))/2$, and because $F_1'(z)$ and $F_2'(z)$ are symmetric and antisymmetric polynomials, respectively.

3.3 Perceptual weighting

The perceptual weighting filter is based on the unquantized LP filter coefficients a_i, and is given by:

$$W(z) = \frac{A(z/\gamma_1)}{A(z/\gamma_2)} = \frac{1 + \sum_{i=1}^{10} \gamma_1^i a_i z^{-i}}{1 + \sum_{i=1}^{10} \gamma_2^i a_i z^{-i}} \tag{27}$$

The values of γ_1 and γ_2 determine the frequency response of the filter $W(z)$. By proper adjustment of these variables it is possible to make the weighting more effective. This is done by making γ_1 and γ_2 a function of the spectral shape of the input signal. This adaptation is done once per 10 ms frame, but an interpolation procedure for each first subframe is used to smooth this adaptation process. The spectral shape is obtained from a 2nd order linear prediction filter, obtained as a by-product from the Levinson-Durbin recursion (3.2.2). The reflection coefficients k_i are converted to Log Area Ratio (LAR) coefficients o_i by:

$$o_i = \log \frac{(1.0 + k_i)}{(1.0 - k_i)} \qquad i = 1, 2 \tag{28}$$

The LAR coefficients corresponding to the current 10 ms frame are used for the second subframe. The LAR coefficients for the first subframe are obtained through linear interpolation with the LAR parameters from the previous frame. The interpolated LAR coefficients in each of the two subframes are given by:

$$\begin{aligned} &\textit{Subframe 1:}\ o_i^{(1)} = 0.5\, o_i^{(previous)} + 0.5\, o_i^{(current)} && i = 1, 2 \\ &\textit{Subframe 2:}\ o_i^{(2)} = o_i^{(current)} && i = 1, 2 \end{aligned} \tag{29}$$

The spectral envelope is characterized as being either flat ($flat = 1$) or tilted ($flat = 0$). For each subframe this characterization is obtained by applying a threshold function to the LAR coefficients. To avoid rapid changes, a hysteresis is used by taking into account the value of *flat* in the previous subframe $m - 1$,

$$flat^{(m)} = \begin{cases} 0 & \text{if } o_1^{(m)} < -1.74 \text{ and } o_2^{(m)} > 0.65 \text{ and } flat^{(m-1)} = 1 \\ 1 & \text{if } \left(o_1^{(m)} > -1.52 \text{ or } o_2^{(m)} < 0.43\right) \text{ and } flat^{(m-1)} = 0 \\ flat^{(m-1)} & \text{otherwise} \end{cases} \tag{30}$$

If the interpolated spectrum for a subframe is classified as flat ($flat^{(m)} = 1$), the weight factors are set to $\gamma_1 = 0.94$ and $\gamma_2 = 0.6$. If the spectrum is classified as tilted ($flat^{(m)} = 0$), the value of γ_1 is set to 0.98, and the value of γ_2 is adapted to the strength of the resonances in the LP synthesis filter, but is bounded between 0.4 and 0.7. If a strong resonance is present, the value of γ_2 is set closer to the upper bound. This adaptation is achieved by a criterion based on the minimum distance between two successive LSP coefficients for the current subframe. The minimum distance is given by:

$$d_{min} = min\,[\omega_{i+1} - \omega_i] \qquad i = 1,...,9 \tag{31}$$

The value of γ_2 is computed using the linear relationship:

$$\gamma_2 = -6.0\, d_{min} + 1.0 \qquad \text{bounded by } 0.4 \le \gamma_2 \le 0.7 \tag{32}$$

The weighted speech signal in a subframe is given by:

$$sw(n) = s(n) + \sum_{i=1}^{10} a_i \gamma_1^i s(n - i) - \sum_{i=1}^{10} a_i \gamma_2^i sw(n - i) \qquad n = 0,...,39 \tag{33}$$

The weighted speech signal $sw(n)$ is used to find an estimation of the pitch delay in the speech frame.

3.4 Open-loop pitch analysis

To reduce the complexity of the search for the best adaptive-codebook delay, the search range is limited around a candidate delay T_{op}, obtained from an open-loop pitch analysis. This open-loop pitch analysis is done once per frame (10 ms). The open-loop pitch estimation uses the weighted speech signal $sw(n)$ of Equation (33), and is done as follows: In the first step, three maxima of the correlation:

$$R(k) = \sum_{n=0}^{79} sw(n)\, sw(n - k) \tag{34}$$

are found in the following three ranges:

$$i = 1: \ 80,...,143$$

$$i = 2: \ 40,...,79$$

$$i = 3: \ 20,...,39$$

The retained maxima $R(t_i)$, $i = 1,...,3$, are normalized through:

$$R'(t_i) = \frac{R(t_i)}{\sqrt{\sum_n sw^2\,(n - t_i)}} \qquad i = 1,...,3 \tag{35}$$

The winner among the three normalized correlations is selected by favouring the delays with the values in the lower range. This is done by weighting the normalized correlations corresponding to the longer delays. The best open-loop delay T_{op} is determined as follows:

$T_{op} = t_1$

$R'(T_{op}) = R'(t_1)$

if $R'(t_2) \geq 0.85R'(T_{op})$

$\quad R'(T_{op}) = R'(t_2)$

$\quad T_{op} = t_2$

end

if $R'(t_3) \geq 0.85R'(T_{op})$

$\quad R'(T_{op}) = R'(t_3)$

$\quad T_{op} = t_3$

end

This procedure of dividing the delay range into three sections and favouring the smaller values is used to avoid choosing pitch multiples.

3.5 Computation of the impulse response

The impulse response $h(n)$ of the weighted synthesis filter $W(z)/\hat{A}(z)$ is needed for the search of adaptive and fixed codebooks. The impulse response $h(n)$ is computed for each subframe by filtering a signal consisting of the coefficients of the filter $A(z/\gamma_1)$ extended by zeros through the two filters $1/\hat{A}(z)$ and $1/A(z/\gamma_2)$.

3.6 Computation of the target signal

The target signal $x(n)$ for the adaptive-codebook search is usually computed by subtracting the zero-input response of the weighted synthesis filter $W(z)/\hat{A}(z) = A(z/\gamma_1)/[\hat{A}(z)A(z/\gamma_2)]$ from the weighted speech signal $sw(n)$ of Equation (33). This is done on a subframe basis.

An equivalent procedure for computing the target signal, which is used in this Recommendation, is the filtering of the LP residual signal $r(n)$ through the combination of synthesis filter $1/\hat{A}(z)$ and the weighting filter $A(z/\gamma_1)/A(z/\gamma_2)$. After determining the excitation for the subframe, the initial states of these filters are updated by filtering the difference between the residual and excitation signals. The memory update of these filters is explained in 3.10.

The residual signal $r(n)$, which is needed for finding the target vector is also used in the adaptive-codebook search to extend the past excitation buffer. This simplifies the adaptive-codebook search procedure for delays less than the subframe size of 40 as will be explained in the next subclause. The LP residual is given by:

$$r(n) = s(n) + \sum_{i=1}^{10} \hat{a}_i s(n - i) \qquad n = 0,...,39 \tag{36}$$

3.7 Adaptive-codebook search

The adaptive-codebook parameters (or pitch parameters) are the delay and gain. In the adaptive-codebook approach for implementing the pitch filter, the excitation is repeated for delays less than the subframe length. In the search stage, the excitation is extended by the LP residual to simplify the closed-loop search. The adaptive-codebook search is done every (5 ms) subframe. In the first subframe, a fractional pitch delay T_1 is used with a resolution of 1/3 in the range of $[\,19\frac{1}{3}, 84\frac{2}{3}\,]$ and integers only in the range [85, 143]. For the second subframe, a delay T_2 with a resolution of 1/3 is always used in the range $int(T_1) - 5\frac{2}{3}$, $int(T_1) + 4\frac{2}{3}$, where $int(T_1)$ is the integer part of the fractional pitch delay T_1 of the first subframe. This range is adapted for the cases where T_1 straddles the boundaries of the delay range.

For each subframe the optimal delay is determined using closed-loop analysis that minimizes the weighted mean-squared error. In the first subframe the delay T_1 is found by searching a small range (six samples) of delay values around the open-loop delay T_{op} (see 3.4). The search boundaries t_{min} and t_{max} are defined by:

$t_{min} = T_{op} - 3$

if $t_{min} < 20$ *then* $t_{min} = 20$

$t_{max} = t_{min} + 6$

if $t_{max} > 143$ *then*

$t_{max} = 143$

$t_{min} = t_{max} - 6$

end

For the second subframe, closed-loop pitch analysis is done around the pitch selected in the first subframe to find the optimal delay T_2. The search boundaries are between $t_{min} - \frac{2}{3}$ and $t_{max} + \frac{2}{3}$, where t_{min} and t_{max} are derived from T_1 as follows:

$t_{min} = int(T_1) - 5$

if $t_{min} < 20$ *then* $t_{min} = 20$

$t_{max} = t_{min} + 9$

if $t_{max} > 143$ *then*

$t_{max} = 143$

$t_{min} = t_{max} - 9$

end

The closed-loop pitch search minimizes the mean-squared weighted error between the original and reconstructed speech. This is achieved by maximizing the term:

$$R(k) = \frac{\sum_{n=0}^{39} x(n)\, y_k(n)}{\sqrt{\sum_{n=0}^{39} y_k(n)\, y_k(n)}} \tag{37}$$

where $x(n)$ is the target signal and $y_k(n)$ is the past filtered excitation at delay k [past excitation convolved with $h(n)$]. Note that the search range is limited around a preselected value, which is the open-loop pitch T_{op} for the first subframe, and T_1 for the second subframe.

The convolution $y_k(n)$ is computed for the delay t_{min}. For the other integer delays in the search range $k = t_{min} + 1,...,t_{max}$, it is updated using the recursive relation:

$$y_k(n) = y_{k-1}(n-1) + u(-k)h(n) \qquad n = 39,...,0 \tag{38}$$

where $u(n)$, $n = -143,...,39$, is the excitation buffer, and $y_{k-1}(-1) = 0$. Note that in the search stage, the samples $u(n)$, $n = 0,...,39$ are not known, and they are needed for pitch delays less than 40. To simplify the search, the LP residual is copied to $u(n)$ to make the relation in Equation (38) valid for all delays.

For the determination of T_2, and T_1 if the optimum integer closed-loop delay is less than 85, the fractions around the optimum integer delay have to be tested. The fractional pitch search is done by interpolating the normalized correlation in Equation (37) and searching for its maximum. The interpolation is done using a FIR filter b_{12} based on a Hamming windowed sinc function with the sinc truncated at ± 11 and padded with zeros at ± 12 ($b_{12}(12) = 0$). The filter has its cut-off frequency (–3 dB) at 3600 Hz in the oversampled domain. The interpolated values of $R(k)$ for the fractions $-\frac{2}{3}$, $-\frac{1}{3}$, 0, $\frac{1}{3}$ and $\frac{2}{3}$ are obtained using the interpolation formula

$$R(k)_t = \sum_{i=0}^{3} R(k-i)b_{12}(t+3i) + \sum_{i=0}^{3} R(k+1+i)b_{12}(3-t+3i) \qquad t = 0, 1, 2 \tag{39}$$

where $t = 0, 1, 2$ corresponds to the fractions 0, $\frac{1}{3}$ and $\frac{2}{3}$, respectively. Note that it is necessary to compute the correlation terms in Equation (37) using a range $t_{min} - 4$, $t_{max} + 4$, to allow for the proper interpolation.

3.7.1 Generation of the adaptive-codebook vector

Once the pitch delay has been determined, the adaptive-codebook vector $v(n)$ is computed by interpolating the past excitation signal $u(n)$ at the given integer delay k and fraction t:

$$v(n) = \sum_{i=0}^{9} u(n-k+i)b_{30}(t+3i) + \sum_{i=0}^{9} u(n-k+1+i)b_{30}(3-t+3i) \quad n = 0,...,39 \quad t = 0, 1, 2 \tag{40}$$

The interpolation filter b_{30} is based on a Hamming windowed sinc functions truncated at ± 29 and padded with zeros at ± 30 [$b_{30}(30) = 0$]. The filter has a cut-off frequency (–3 dB) at 3600 Hz in the oversampled domain.

3.7.2 Codeword computation for adaptive-codebook delays

The pitch delay T_1 is encoded with 8 bits in the first subframe and the relative delay in the second subframe is encoded with 5 bits. A fractional delay T is represented by its integer part $int(T)$, and a fractional part $frac/3$, $frac = -1,0,1$. The pitch index $P1$ is now encoded as:

$$P1 = \begin{cases} 3(int(T_1) - 19) + frac - 1 & \text{if } T_1 = [19,...,85],\ frac = [-1, 0, 1] \\ (int(T_1) - 85) + 197 & \text{if } T_1 = [86,...,143],\ frac = 0 \end{cases} \tag{41}$$

The value of the pitch delay T_2 is encoded relative to the value of T_1. Using the same interpretation as before, the fractional delay T_2 represented by its integer part $\text{int}(T_2)$, and a fractional part $frac/3$, $frac = -1,0,1$, is encoded as:

$$P2 = 3(int(T_2) - t_{min}) + frac + 2 \tag{42}$$

where t_{min} is derived from T_1 as in 3.7.

To make the coder more robust against random bit errors, a parity bit $P0$ is computed on the delay index $P1$ of the first subframe. The parity bit is generated through an XOR operation on the six most significant bits of $P1$. At the decoder this parity bit is recomputed and if the recomputed value does not agree with the transmitted value, an error concealment procedure is applied.

3.7.3 Computation of the adaptive-codebook gain

Once the adaptive-codebook delay is determined, the adaptive-codebook gain g_p is computed as:

$$g_p = \frac{\sum_{n=0}^{39} x(n)\, y(n)}{\sum_{n=0}^{39} y(n)\, y(n)} \qquad \text{bounded by } 0 \le g_p \le 1.2 \tag{43}$$

where $x(n)$ is the target signal and $y(n)$ is the filtered adaptive-codebook vector [zero-state response of $W(z)/\hat{A}(z)$ to $v(n)$]. This vector is obtained by convolving $v(n)$ with $h(n)$:

$$y(n) = \sum_{i=0}^{n} v(i)h(n - i) \quad n = 0,...,39 \tag{44}$$

3.8 Fixed codebook – Structure and search

The fixed codebook is based on an algebraic codebook structure using an Interleaved Single-Pulse Permutation (ISPP) design. In this codebook, each codebook vector contains four non zero pulses. Each pulse can have either the amplitudes +1 or −1, and can assume the positions given in Table 7.

TABLE 7/G.729

Structure of fixed codebook $\mathcal{C}$

Pulse	Sign	Positions
i_0	s_0: ±1	m_0: 0, 5, 10, 15, 20, 25, 30, 35
i_1	s_1: ±1	m_1: 1, 6, 11, 16, 21, 26, 31, 36
i_2	s_2: ±1	m_2: 2, 7, 12, 17, 22, 27, 32, 37
i_3	s_3: ±1	m_3: 3, 8, 13, 18, 23, 28, 33, 38 4, 9, 14, 19, 24, 29, 34, 39

The codebook vector $c(n)$ is constructed by taking a zero vector of dimension 40, and putting the four unit pulses at the found locations, multiplied with their corresponding sign:

$$c(n) = s_0\delta(n - m_0) + s_1\delta(n - m_1) + s_2\delta(n - m_2) + s_3\delta(n - m_3) \quad n = 0,...,39 \tag{45}$$

where $\delta(0)$ is a unit pulse. A special feature incorporated in the codebook is that the selected codebook vector is filtered through an adaptive pre-filter $P(z)$ that enhances harmonic components to improve the quality of the reconstructed speech. Here the filter:

$$P(z) = 1 / (1 - \beta z^{-T}) \tag{46}$$

is used, where T is the integer component of the pitch delay of the current subframe, and β is a pitch gain. The value of β is made adaptive by using the quantized adaptive-codebook gain from the previous subframe, that is:

$$\beta = \hat{g}_p^{(m-1)} \qquad \text{bounded by } 0.2 \le \beta \le 0.8 \tag{47}$$

For delays less than 40, the codebook $c(n)$ of Equation (45) is modified according to:

$$c(n) = \begin{cases} c(n) & n = 0,...,T - 1 \\ c(n) + \beta c(n - T) & n = T,...,39 \end{cases} \tag{48}$$

This modification is incorporated in the fixed-codebook search by modifying the impulse response $h(n)$ according to:

$$h(n) = \begin{cases} h(n) & n = 0,...,T - 1 \\ h(n) + \beta h(n - T) & n = T,...,39 \end{cases} \tag{49}$$

3.8.1 Fixed-codebook search procedure

The fixed codebook is searched by minimizing the mean-squared error between the weighted input speech $sw(n)$ of Equation (33) and the weighted reconstructed speech. The target signal used in the closed-loop pitch search is updated by subtracting the adaptive-codebook contribution. That is:

$$x'(n) = x(n) - g_p y(n) \qquad n = 0,...,39 \tag{50}$$

where $y(n)$ is the filtered adaptive-codebook vector of Equation (44) and g_p the adaptive-codebook gain of Equation (43).

The matrix **H** is defined as the lower triangular Toepliz convolution matrix with diagonal $h(0)$ and lower diagonal $h(1),...,h(39)$. The matrix $\Phi = \mathbf{H}^t\mathbf{H}$ contains the correlations of $h(n)$, and the elements of this symmetric matrix are given by:

$$\phi(i,j) = \sum_{n=j}^{39} h(n-i)h(n-j) \qquad i = 0,...,39 \quad j = i,...,39 \tag{51}$$

The correlation signal $d(n)$ is obtained from the target signal $x'(n)$ and the impulse response $h(n)$ by:

$$d(n) = \sum_{i=n}^{39} x'(i)h(i-n) \qquad n = 0,...,39 \tag{52}$$

If c_k is the kth fixed-codebook vector, then the codebook is search by maximizing the term:

$$\frac{C_k^2}{E_k} = \frac{\left(\sum_{n=0}^{39} d(n)c_k(n)\right)^2}{c_k^t \Phi c_k} \tag{53}$$

where t denotes transpose.

The signal $d(n)$ and the matrix Φ are computed before the codebook search. Note that only the elements actually needed are computed and an efficient storage procedure has been designed to speed up the search procedure.

The algebraic structure of the codebook $\mathcal{C}$ allows for a fast search procedure since the codebook vector c_k contains only four non zero pulses. The correlation in the numerator of Equation (53) for a given vector c_k is given by:

$$C = \sum_{i=0}^{3} s_i d(m_i) \tag{54}$$

where m_i is the position of the ith pulse and s_i is its amplitude. The energy in the denominator of Equation (53) is given by:

$$E = \sum_{i=0}^{3} \phi(m_i, m_i) + 2 \sum_{i=0}^{2} \sum_{j=i+1}^{3} s_i s_j \phi(m_i, m_j) \tag{55}$$

To simplify the search procedure, the pulse amplitudes are predetermined by quantizing the signal $d(n)$. This is done by setting the amplitude of a pulse at a certain position equal to the sign of $d(n)$ at the position. Before the codebook search, the following steps are done. First, the signal $d(n)$ is decomposed into two parts: its absolute value $|d(n)|$ and its sign $sign\ [d(n)]$. Second, the matrix Φ is modified by including the sign information; that is,

$$\phi'(i,j) = sign\ [d(i)]\ sign\ [d(j)]\phi(i,j) \qquad i = 0,...,39 \quad j = i+1,...,39 \tag{56}$$

The main-diagonal elements of Φ are scaled to remove the factor 2 in Equation (55)

$$\phi'(i,i) = 0.5\phi'(i,i) \qquad i = 0,...,39 \tag{57}$$

The correlation in Equation (54) is now given by:

$$C = |d(m_0)| + |d(m_1)| + |d(m_2)| + |d(m_3)| \tag{58}$$

and the energy in Equation (55) is given by:

$$\begin{aligned} E/2 &= \phi'(m_0, m_0) \\ &+ \phi'(m_1, m_1) + \phi'(m_0, m_1) \\ &+ \phi'(m_2, m_2) + \phi'(m_0, m_2) + \phi'(m_1, m_2) \\ &+ \phi'(m_3, m_3) + \phi'(m_0, m_3) + \phi'(m_1, m_3) + \phi'(m_2, m_3) \end{aligned} \tag{59}$$

A focused search approach is used to further simplify the search procedure. In this approach a precomputed threshold is tested before entering the last loop, and the loop is entered only if this threshold is exceeded. The maximum number of times the loop can be entered is fixed so that a low percentage of the codebook is searched. The threshold is computed based on the correlation C. The maximum absolute correlation and the average correlation due to the contribution of the first three pulses, max_3 and av_3, are found before the codebook search. The threshold is given by:

$$thr_3 = av_3 + K_3(max_3 - av_3) \tag{60}$$

The fourth loop is entered only if the absolute correlation (due to three pulses) exceeds thr_3, where $0 \leq K_3 < 1$. The value of K_3 controls the percentage of codebook search and it is set here to 0.4. Note that this results in a variable search time. To further control the search the number of times the last loop is entered (for the two subframes) cannot exceed a certain maximum, which is set here to 180 (the average worst case per subframe is 90 times).

3.8.2 Codeword computation of the fixed codebook

The pulse positions of the pulses i_0, i_1 and i_2, are encoded with 3 bits each, while the position of i_3 is encoded with 4 bits. Each pulse amplitude is encoded with 1 bit. This gives a total of 17 bits for the 4 pulses. By defining $s = 1$ if the sign is positive and $s = 0$ if the sign is negative, the sign codeword is obtained from:

$$S = s_0 + 2s_1 + 4s_2 + 8s_3 \tag{61}$$

and the fixed-codebook codeword is obtained from:

$$C = (m_0/5) + 8(m_1/5) + 64(m_2/5) + 512(2(m_3/5) + jx) \tag{62}$$

where $jx = 0$ if $m_3 = 3,8,...,38$, and $jx = 1$ if $m_3 = 4,9...,39$.

3.9 Quantization of the gains

The adaptive-codebook gain (pitch gain) and the fixed-codebook gain are vector quantized using 7 bits. The gain codebook search is done by minimizing the mean-squared weighted error between original and reconstructed speech which is given by:

$$E = \mathbf{x}^t\mathbf{x} + g_p^2\,\mathbf{y}^t\mathbf{y} + g_c^2\,\mathbf{z}^t\mathbf{z} - 2g_p\mathbf{x}^t\mathbf{y} - 2g_c\mathbf{x}^t\mathbf{z} + 2g_pg_c\mathbf{y}^t\mathbf{z} \tag{63}$$

where $\mathbf{x}$ is the target vector (see 3.6), $\mathbf{y}$ is the filtered adaptive-codebook vector of Equation (44), and $\mathbf{z}$ is the fixed-codebook vector convolved with $h(n)$,

$$z(n) = \sum_{i=0}^{n} c(i)h(n - i) \qquad n = 0,...,39 \tag{64}$$

3.9.1 Gain prediction

The fixed-codebook gain g_c can be expressed as:

$$g_c = \gamma g_c' \tag{65}$$

where g_c' is a predicted gain based on previous fixed-codebook energies, and γ is a correction factor.

The mean energy of the fixed-codebook contribution is given by:

$$E = 10 \log \left(\frac{1}{40} \sum_{n=0}^{39} c(n)^2 \right) \tag{66}$$

After scaling the vector $c(n)$ with the fixed-codebook gain g_c, the energy of the scaled fixed codebook is given by 20 log g_c + E. Let $E^{(m)}$ be the mean-removed energy (in dB) of the (scaled) fixed-codebook contribution at the subframe m, given by:

$$E^{(m)} = 20 \log g_c + E - \bar{E} \tag{67}$$

where $\bar{E}$ = 30 dB is the mean energy of the fixed-codebook excitation. The gain g_c can be expressed as a function of $E^{(m)}$, E and $\bar{E}$ by:

$$g_c = 10^{(E^{(m)} + \bar{E} - E)/20} \tag{68}$$

The predicted gain g_c' is found by predicting the log-energy of the current fixed-codebook contribution from the log-energy of previous fixed-codebook contributions. The 4th order MA prediction is done as follows. The predicted energy is given by:

$$\tilde{E}^{(m)} = \sum_{i=1}^{4} b_i \hat{U}^{(m-i)} \tag{69}$$

where $[b_1\ b_2\ b_3\ b_4]$ = [0.68 0.58 0.34 0.19] are the MA prediction coefficients, and $\hat{U}^{(m)}$ is the quantized version of the prediction error $U^{(m)}$ at subframe m, defined by:

$$U^{(m)} = E^{(m)} - \tilde{E}^{(m)} \tag{70}$$

The predicted gain g_c' is found by replacing $E^{(m)}$ by its predicted value in Equation (68).

$$g_c' = 10^{(\tilde{E}^{(m)} + \bar{E} - E)/20} \tag{71}$$

The correction factor γ is related to the gain-prediction error by:

$$U^{(m)} = E^{(m)} - \tilde{E}^{(m)} = 20 \log(\gamma) \tag{72}$$

3.9.2 Codebook search for gain quantization

The adaptive-codebook gain, g_p, and the factor γ are vector quantized using a two-stage conjugate structured codebook. The first stage consists of a 3 bit two-dimensional codebook $\mathcal{GA}$, and the second stage consists of a 4 bit two-dimensional codebook $\mathcal{GB}$. The first element in each codebook represents the quantized adaptive-codebook gain $\hat{g}_p$, and the second element represents the quantized fixed-codebook gain correction factor $\hat{\gamma}$. Given codebook indices GA and GB for $\mathcal{GA}$ and $\mathcal{GB}$, respectively, the quantized adaptive-codebook gain is given by:

$$\hat{g}_p = \mathcal{GA}_1(GA) + \mathcal{GB}_1(GB) \tag{73}$$

and the quantized fixed-codebook gain by:

$$\hat{g}_c = g'_c\hat{\gamma} = g'_c(\mathcal{GA}_2(GA) + \mathcal{GB}_2(GB)) \tag{74}$$

This conjugate structure simplifies the codebook search, by applying a pre-selection process. The optimum pitch gain g_p, and fixed-codebook gain, g_c, are derived from Equation (63), and are used for the pre-selection. The codebook $\mathcal{GA}$ contains eight entries in which the second element (corresponding to g_c) has in general larger values than the first element (corresponding to g_p). This bias allows a pre-selection using the value of g_c. In this pre-selection process, a cluster of four vectors whose second elements are close to g_c are selected. Similarly, the codebook $\mathcal{GB}$ contains 16 entries in which each has a bias towards the first element (corresponding to g_p). A cluster of eight vectors whose first elements are close to g_p are selected. Hence for each codebook the best 50% candidate vectors are selected. This is followed by an exhaustive search over the remaining $4 \times 8 = 32$ possibilities, such that the combination of the two indices minimizes the weighted mean-squared error of Equation (63).

3.9.3 Codeword computation for gain quantizer

The codewords *GA* and *GB* for the gain quantizer are obtained from the indices corresponding to the best choice. To reduce the impact of single bit errors the codebook indices are mapped.

3.10 Memory update

An update of the states of the synthesis and weighting filters is needed to compute the target signal in the next subframe. After the two gains are quantized, the excitation signal, $u(n)$, in the present subframe is obtained using:

$$u(n) = \hat{g}_p v(n) + \hat{g}_c c(n) \qquad n = 0,...,39 \tag{75}$$

where $\hat{g}_p$ and $\hat{g}_c$ are the quantized adaptive and fixed-codebook gains, respectively, $v(n)$ is the adaptive-codebook vector (interpolated past excitation), and $c(n)$ is the fixed-codebook vector including harmonic enhancement. The states of the filters can be updated by filtering the signal $r(n)$-$u(n)$ (difference between residual and excitation) through the filters $1/\hat{A}(z)$ and $A(z/\gamma_1)/A(z/\gamma_2)$ for the 40 sample subframe and saving the states of the filters. This would require three filter operations. A simpler approach, which requires only one filter operation, is as follows. The locally reconstructed speech $\hat{s}(n)$ is computed by filtering the excitation signal through $1/\hat{A}(z)$. The output of the filter due to the input $r(n) - u(n)$ is equivalent to $e(n) = s(n) - \hat{s}(n)$. So the states of the synthesis filter $1/\hat{A}(z)$ are given by $e(n)$, $n = 30,...,39$. Updating the states of the filter $A(z/\gamma_1)/A(z/\gamma_2)$ can be done by filtering the error signal $e(n)$ through this filter to find the perceptually weighted error $ew(n)$. However, the signal $ew(n)$ can be equivalently found by:

$$ew(n) = x(n) - \hat{g}_p y(n) - \hat{g}_c z(n) \tag{76}$$

Since the signals $x(n)$, $y(n)$ and $z(n)$ are available, the states of the weighting filter are updated by computing $ew(n)$ as in Equation (76) for $n = 30,...,39$. This saves two filter operations.

4 Functional description of the decoder

The principle of the decoder was shown in clause 2 (Figure 3). First the parameters are decoded (LP coefficients, adaptive-codebook vector, fixed-codebook vector and gains). The transmitted parameters are listed in Table 8. These decoded parameters are used to compute the reconstructed speech signal as will be described in 4.1. This reconstructed signal is enhanced by a post-processing operation consisting of a postfilter, a high-pass filter and an upscaling (see 4.2). Subclause 4.4 describes the error concealment procedure used when either a parity error has occurred, or when the frame erasure flag has been set. A detailed signal flow diagram of the decoder is shown in Figure 6.

TABLE 8/G.729

Description of transmitted parameters indices – The bitstream ordering is reflected by the order in the table – For each parameter the Most Significant Bit (MSB) is transmitted first

Symbol	Description	Bits
L0	Switched MA predictor of LSP quantizer	1
L1	First stage vector of quantizer	7
L2	Second stage lower vector of LSP quantizer	5
L3	Second stage higher vector of LSP quantizer	5
P1	Pitch delay first subframe	8
P0	Parity bit for pitch delay	1
C1	Fixed codebook first subframe	13
S1	Signs of fixed-codebook pulses 1st subframe	4
GA1	Gain codebook (stage 1) 1st subframe	3
GB1	Gain codebook (stage 2) 1st subframe	4
P2	Pitch delay second subframe	5
C2	Fixed codebook 2nd subframe	13
S2	Signs of fixed-codebook pulses 2nd subframe	4
GA2	Gain codebook (stage 1) 2nd subframe	3
GB2	Gain codebook (stage 2) 2nd subframe	4

4.1 Parameter decoding procedure

The decoding process is done in the following order.

4.1.1 Decoding of LP filter parameters

The received indices *L0*, *L1*, *L2* and *L3* of the LSP quantizer are used to reconstruct the quantized LSP coefficients using the procedure described in 3.2.4. The interpolation procedure described in 3.2.5 is used to obtain two sets of interpolated LSP coefficients (corresponding to two subframes). For each subframe, the interpolated LSP coefficients are converted to LP filter coefficients a_i, which are used for synthesizing the reconstructed speech in the subframe.

The following steps are repeated for each subframe:

1) decoding of the adaptive-codebook vector;

2) decoding of the fixed-codebook vector;

3) decoding of the adaptive and fixed-codebook gains;

4) computation of the reconstructed speech.

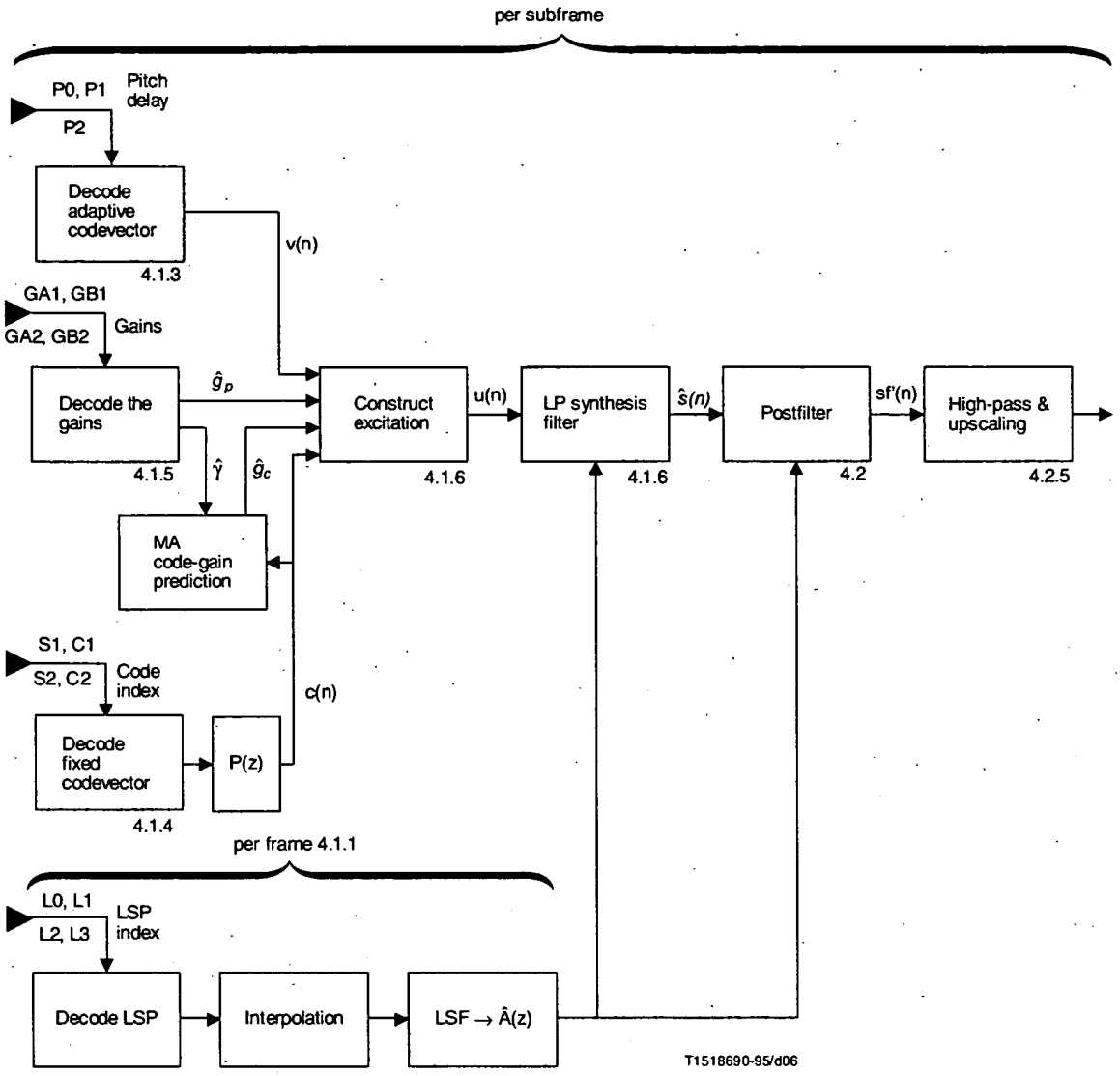

FIGURE 6/G.729

Signal flow at the CS-ACELP decoder

4.1.2 Computation of the parity bit

Before the excitation is reconstructed, the parity bit is recomputed from the adaptive-codebook delay index $P1$ (see 3.7.2). If this bit is not identical to the transmitted parity bit $P0$, it is likely that bit errors occurred during transmission.

If a parity error occurs on $P1$, the delay value T_1 is set to the integer part of the delay value T_2 of the previous frame. The value T_2 is derived with the procedure outlined in 4.1.3, using this new value of T_1.

4.1.3 Decoding of the adaptive-codebook vector

If no parity error has occurred the received adaptive-codebook index $P1$ is used to find the integer and fractional parts of the pitch delay T_1. The integer part $int(T_1)$ and fractional part $frac$ of T_1 are obtained from $P1$ as follows:

if $P1 < 197$

$int(T_1) = (P1 + 2)/3 + 19$

$frac = P1 - 3\, int(T_1) + 58$

else

$int(T_1) = P1 - 112$

$frac = 0$

end

The integer and fractional part of T_2 are obtained from $P2$ and t_{min}, where t_{min} is derived from T_1 as follows:

$t_{min} = int(T_1) - 5$

if $t_{min} < 20$ *then* $t_{min} = 20$

$t_{max} = t_{min} + 9$

if $t_{max} > 143$ *then*

$t_{max} = 143$

$t_{min} = t_{max} - 9$

end

Now T_2 is decoded using:

$$int(T_2) = (P2 + 2)/3 - 1 + t_{min}$$

$$frac = P2 - 2 - 3\,((P2 + 2)/3 - 1)$$

The adaptive-codebook vector $v(n)$ is found by interpolating the past excitation $u(n)$ (at the pitch delay) using Equation (40).

4.1.4 Decoding of the fixed-codebook vector

The received fixed-codebook index C is used to extract the positions of the excitation pulses. The pulse signs are obtained from S. This is done by reversing the process described in 3.8.2. Once the pulse positions and signs are decoded the fixed-codebook vector $c(n)$ is constructed using Equation (45). If the integer part of the pitch delay T is less than the subframe size 40, $c(n)$ is modified according to Equation (48).

4.1.5 Decoding of the adaptive and fixed-codebook gains

The received gain-codebook index gives the adaptive-codebook gain $\hat{g}_p$ and the fixed-codebook gain correction factor $\hat{\gamma}$. This procedure is described in detail in 3.9. The estimated fixed-codebook gain g'_c is found using Equation (71). The fixed-codebook vector is obtained from the product of the quantized gain correction factor with this predicted gain Equation (74). The adaptive-codebook gain is reconstructed using Equation (73).

4.1.6 Computing the reconstructed speech

The excitation $u(n)$ [see Equation (75)] is input to the LP synthesis filter. The reconstructed speech for the subframe is given by:

$$\hat{s}(n) = u(n) - \sum_{i=1}^{10} \hat{a}_i \hat{s}(n - i) \qquad n = 0,...,39 \tag{77}$$

where $\hat{a}_i$ are the interpolated LP filter coefficients for the current subframe. The reconstructed speech $\hat{s}(n)$ is then processed by the post processor described in the next subclause.

4.2 Post-processing

Post-processing consists of three functions: adaptive postfiltering, high-pass filtering and signal upscaling. The adaptive postfilter is the cascade of three filters: a long-term postfilter $H_p(z)$, a short-term postfilter $H_f(z)$ and a tilt compensation filter $H_t(z)$, followed by an adaptive gain control procedure. The postfilter coefficients are updated every 5 ms subframe. The postfiltering process is organized as follows. First, the reconstructed speech $\hat{s}(n)$ is inverse filtered through $\hat{A}(z/\gamma_n)$ to produce the residual signal $\hat{r}(n)$. This signal is used to compute the delay T and gain g_l of the long-term postfilter $H_p(z)$. The signal $\hat{r}(n)$ is then filtered through the long-term postfilter $H_p(z)$ and the synthesis filter $1/[g_f\hat{A}(z/\gamma_d)]$. Finally, the output signal of the synthesis filter $1/[g_f\hat{A}(z/\gamma_d)]$ is passed through the tilt compensation filter $H_t(z)$ to generate the postfiltered reconstructed speech signal $sf(n)$. Adaptive gain control is then applied to $sf(n)$ to match the energy of $\hat{s}(n)$. The resulting signal $sf'(n)$ is high-pass filtered and scaled to produce the output signal of the decoder.

4.2.1 Long-term postfilter

The long-term postfilter is given by:

$$H_p(z) = \frac{1}{1 + \gamma_p g_l}(1 + \gamma_p g_l z^{-T}) \tag{78}$$

where T is the pitch delay, and g_l is the gain coefficient. Note that g_l is bounded by 1, and it is set to zero if the long-term prediction gain is less than 3 dB. The factor γ_p controls the amount of long-term postfiltering and has the value of $\gamma_p = 0.5$. The long-term delay and gain are computed from the residual signal $\hat{r}(n)$ obtained by filtering the speech $\hat{s}(n)$ through $\hat{A}(z/\gamma_n)$, which is the numerator of the short-term postfilter (see 4.2.2).

$$\hat{r}(n) = \hat{s}(n) + \sum_{i=1}^{10} \gamma_n^i \hat{a}_i \hat{s}(n - i) \tag{79}$$

The long-term delay is computed using a two-pass procedure. The first pass selects the best integer T_0 in the range $[int(T_1) - 1, int(T_1) + 1]$, where $int(T_1)$ is the integer part of the (transmitted) pitch delay T_1 in the first subframe. The best integer delay is the one that maximizes the correlation.

$$R(k) = \sum_{n=0}^{39} \hat{r}(n)\hat{r}(n - k) \tag{80}$$

The second pass chooses the best fractional delay T with resolution 1/8 around T_0. This is done by finding the delay with the highest pseudo-normalized correlation

$$R'(k) = \frac{\sum_{n=0}^{39} \hat{r}(n)\hat{r}_k(n)}{\sqrt{\sum_{n=0}^{39} \hat{r}_k(n)\hat{r}_k(n)}} \tag{81}$$

where $\hat{r}_k(n)$ is the residual signal at delay k. Once the optimal delay T is found, the corresponding correlation $R'(T)$ is normalized with the square-root of the energy of $\hat{r}(n)$. The squared value of this normalized correlation is used to determine if the long-term postfilter should be disabled. This is done by setting $g_l = 0$ if:

$$\frac{R'(T)^2}{\sum_{n=0}^{39} \hat{r}(n)\hat{r}(n)} < 0.5 \tag{82}$$

Otherwise the value of g_l is computed from:

$$g_l = \frac{\sum_{n=0}^{39} \hat{r}(n)\hat{r}_k(n)}{\sum_{n=0}^{39} \hat{r}_k(n)\hat{r}_k(n)} \qquad \text{bounded by } 0 \leq g_l \leq 1.0 \tag{83}$$

The non-integer delayed signal $\hat{r}_k(n)$ is first computed using an interpolation filter of length 33. After the selection of T, $\hat{r}_k(n)$ is recomputed with a longer interpolation filter of length 129. The new signal replaces the previous one only if the longer filter increases the value of $R'(T)$.

4.2.2 Short-term postfilter

The short-term postfilter is given by:

$$H_f(z) = \frac{1}{g_f} \frac{\hat{A}(z/\gamma_n)}{\hat{A}(z/\gamma_d)} = \frac{1}{g_f} \frac{1 + \sum_{i=1}^{10} \gamma_n^i \hat{a}_i z^{-i}}{1 + \sum_{i=1}^{10} \gamma_d^i \hat{a}_i z^{-i}} \tag{84}$$

where $\hat{A}(z)$ is the received quantized LP inverse filter (LP analysis is not done at the decoder) and the factors γ_n and γ_d control the amount of short-term postfiltering, and are set to $\gamma_n = 0.55$, and $\gamma_d = 0.7$. The gain term g_f is calculated on the truncated impulse response $h_f(n)$ of the filter $\hat{A}(z/\gamma_n)/\hat{A}(z/\gamma_d)$ and is given by:

$$g_f = \sum_{n=0}^{19} |h_f(n)| \tag{85}$$

4.2.3 Tilt compensation

The filter $H_t(z)$ compensates for the tilt in the short-term postfilter $H_f(z)$ and is given by:

$$H_t(z) = \frac{1}{g_t}(1 + \gamma_t k_1' z^{-1}) \tag{86}$$

where $\gamma_t k_1'$ is a tilt factor k_1' being the first reflection coefficient calculated from $h_f(n)$ with

$$k_1' = -\frac{r_h(1)}{r_h(0)} \qquad r_h(i) = \sum_{j=0}^{19-i} h_f(j)\, h_f(j+i) \tag{87}$$

The gain term $g_t = 1 - |\gamma_t k_1'|$ compensates for the decreasing effect of g_f in $H_f(z)$. Furthermore, it has been shown that the product filter $H_f(z)H_t(z)$ has generally no gain. Two values for γ_t are used depending on the sign of k_1'. If k_1' is negative, $\gamma_t = 0.9$, and if k_1' is positive, $\gamma_t = 0.2$.

4.2.4 Adaptive gain controll

Adaptive gain control is used to compensate for gain differences between the reconstructed speech signal $\hat{s}(n)$ and the postfiltered signal $sf(n)$. The gain scaling factor G for the present subframe is computed by:

$$G = \frac{\sum_{n=0}^{39} |\hat{s}(n)|}{\sum_{n=0}^{39} |sf(n)|} \tag{88}$$

The gain-scaled postfiltered signal $sf'(n)$ is given by:

$$sf'(n) = g^{(n)} sf(n) \qquad n = 0,...,39 \tag{89}$$

where $g^{(n)}$ is updated on a sample-by-sample basis and given by:

$$g^{(n)} = 0.85\, g^{(n-1)} + 0.15\, G \qquad n = 0,...,39 \tag{90}$$

The initial value of $g^{(-1)} = 1.0$ is used. Then for each new subframe, $g^{(-1)}$ is set equal to $g^{(39)}$ of the previous subframe.

4.2.5 High-pass filtering and upscaling

A high-pass filter with a cut-off frequency of 100 Hz is applied to the reconstructed postfiltered speech $sf'(n)$. The filter is given by:

$$H_{h2}(z) = \frac{0.93980581 - 1.8795834z^{-1} + 0.93980581z^{-2}}{1 - 1.9330735z^{-1} + 0.93589199z^{-2}} \tag{91}$$

The filtered signal is multiplied by a factor 2 to restore the input signal level.

4.3 Encoder and decoder initialization

All static encoder and decoder variables should be initialized to 0, except the variables listed in Table 9.

TABLE 9/G.729

Description of parameters with non-zero initialization

Variable	Reference	Initial value
β	3.8	0.8
$g^{(-1)}$	4.2.4	1.0
$\hat{l}_i$	3.2.4	$i\pi/11$
q_i	3.2.4	$\arccos(i\pi/11)$
$\hat{U}^{(k)}$	3.9.1	−14

4.4 Concealment of frame erasures

An error concealment procedure has been incorporated in the decoder to reduce the degradation in the reconstructed speech because of frame erasures in the bitstream. This error concealment process is functional when the frame of coder parameters (corresponding to a 10 ms frame) has been identified as being erased. The mechanism for detecting frame erasures is not defined in the Recommendation, and will depend on the application.

The concealment strategy has to reconstruct the current frame, based on previously received information. The method replaces the missing excitation signal with one of similar characteristics, while gradually decaying its energy. This is done by using a voicing classifier based on the long-term prediction gain, which is computed as part of the long-term postfilter analysis. The long-term postfilter (see 4.2.1) finds the long-term predictor for which the prediction gain is more than 3 dB. This is done by setting a threshold of 0.5 on the squared normalized correlation of (Equation 82). For the error concealment process, a 10 ms frame is declared periodic if at least one 5 ms subframe has a long-term prediction gain of more than 3 dB. Otherwise the frame is declared non-periodic. An erased frame inherits its class from the preceding (reconstructed) speech frame. Note that the voicing classification is continuously updated based on this reconstructed speech signal.

The specific steps taken for an erased frame are:

1) repetition of the synthesis filter parameters;

2) attenuation of adaptive and fixed-codebook gains;

3) attenuation of the memory of the gain predictor;

4) generation of the replacement excitation.

4.4.1 Repetition of synthesis filter parameters

The synthesis filter in an erased frame uses the LP parameters of the last good frame. The memory of the MA LSF predictor contains the values of the received codewords $\hat{l}_i$. Since the codeword is not available for the current frame m, it is computed from the repeated LSF parameters $\hat{\omega}_i$ and the predictor memory using:

$$\hat{l}_i = \left[\hat{\omega}_i^{(m)} - \sum_{k=1}^{4} \hat{p}_{i,k}\, \hat{l}_i^{(m-k)}\right] / \left(1 - \sum_{k=1}^{4} \hat{p}_{i,k}\right) \qquad i = 1,...,10 \tag{92}$$

where the MA predictor coefficients $\hat{p}_{i,k}$ are those of the last received good frame.

4.4.2 Attenuation of adaptive and fixed-codebook gains

The fixed-codebook gain is based on an attenuated version of the previous fixed-codebook gain and is given by:

$$g_c^{(m)} = 0.98 g_c^{(m-1)} \tag{93}$$

where m is the subframe index. The adaptive-codebook gain is based on an attenuated version of the previous adaptive-codebook gain and is given by:

$$g_p^{(m)} = 0.9 g_p^{(m-1)} \qquad \text{bounded by } g_p^{(m)} < 0.9 \tag{94}$$

4.4.3 Attenuation of the memory of the gain predictor

As was described in 3.9 the gain predictor uses the energy of previously selected fixed-codebook vectors c(n). To avoid transitional effects at the decoder, once good frames are received, the memory of the gain predictor is updated with an attenuated version of the codebook energy. The value of $\hat{U}^{(m)}$ for the current subframe m is set to the averaged quantized gain prediction-error, attenuated by 4 dB:

$$\hat{U}^{(m)} = \left(0.25 \sum_{i=1}^{4} \hat{U}^{(m-i)}\right) - 4.0 \text{ bounded by } \hat{U}^{(m)} \geq -14 \tag{95}$$

4.4.4 Generation of the replacement excitation

The excitation used depends on the periodicity classification. If the last reconstructed frame was classified as periodic, the current frame is considered to be periodic as well. In that case only the adaptive codebook is used, and the fixed-codebook contribution is set to zero. The pitch delay is based on the integer part of the pitch delay in the previous frame, and is repeated for each successive frame. To avoid excessive periodicity the delay is increased by one for each next subframe but bounded by 143. The adaptive-codebook gain is based on an attenuated value according to Equation (94).

If the last reconstructed frame was classified as non-periodic, the current frame is considered to be non-periodic as well, and the adaptive-codebook contribution is set to zero. The fixed-codebook contribution is generated by randomly selecting a codebook index and sign index. The random generator is based on the function

$$seed = 31821\ seed + 13849 \tag{96}$$

with the initial seed value of 21845. The fixed-codebook index is derived from the 13 least significant bits of the next random number. The fixed-codebook sign is derived from the 4 least significant bits of the next random number. The fixed-codebook gain is attenuated according to Equation (93).

5 Bit-exact description of the CS-ACELP coder

ANSI C code simulating the CS-ACELP coder in 16 bit fixed-point is available from ITU-T. The following subclauses summarize the use of this simulation code, and how the software is organized.

5.1 Use of the simulation software

The C code consists of two main programs **coder.c**, which simulates the encoder, and **decoder.c**, which simulates the decoder. The encoder is run as follows:

coder inputfile bitstreamfile

The input file and output file are sampled data files containing 16 bit PCM signals. The decoder is run as follows:

decoder bitstreamfile outputfile

The mapping table of the encoded bitstream is contained in the simulation software.

5.2 Organization of the simulation software

In the fixed-point ANSI C simulation, only two types of fixed-point data are used as is shown in Table 10. To facilitate the implementation of the simulation code, loop indices, Boolean values and flags use the type **Flag**, which would be either 16 bits or 32 bits depending on the target platform.

TABLE 10/G.729

Data types used in ANSI C simulation

Type	Maximal value	Minimal value	Description
Word16	0x7fff	0x8000	Signed 2's complement 16-bit word
Word32	0x7fffffffL	0x80000000L	Signed 2's complement 32-bit word

All the computations are done using a predefined set of basic operators. The description of these operators is given in Table 11. The tables used by the simulation coder are summarized in Table 12. These main programmes use a library of routines that are summarized in Tables 13, 14 and 15.

TABLE 11/G.729

Basic operations used in ANSI C simulation

Operation	Description
Word16 sature(Word32 L_var1)	Limit to 16 bits
Word16 add(Word16 var1, Word16 var2)	Short addition
Word16 sub(Word16 var1, Word16 var2)	Short subtraction
Word16 abs_s(Word16 var1)	Short absolute value
Word16 shl(Word16 var1, Word16 var2)	Short shift left
Word16 shr(Word16 var1, Word16 var2)	Short shift right
Word16 mult(Word16 var1, Word16 var2)	Short multiplication
Word32 L_mult(Word16 var1, Word16 var2)	Long multiplication
Word16 negate(Word16 var1)	Short negate
Word16 extract_h(Word32 L_var1)	Extract high
Word16 extract_l(Word32 L_var1)	Extract low
Word16 round(Word32 L_var1)	Round
Word32 L_mac(Word32 L_var3, Word16 var1, Word16 var2)	Multiply and accumulate
Word32 L_msu(Word32 L_var3, Word16 var1, Word16 var2)	Multiply and subtract
Word32 L_add(Word32 L_var1, Word32 L_var2)	Long addition
Word32 L_sub(Word32 L_var1, Word32 L_var2)	Long subtraction
Word32 L_negate(Word32 L_var1)	Long negate
Word16 mult_r(Word16 var1, Word16 var2)	Multiplication with rounding
Word32 L_shl(Word32 L_var1, Word16 var2)	Long shift left
Word32 L_shr(Word32 L_var1, Word16 var2)	Long shift right
Word16 shr_r(Word16 var1, Word16 var2)	Shift right with rounding
Word16 mac_r(Word32 L_var3, Word16 var1, Word16 var2)	Mac with rounding
Word16 msu_r(Word32 L_var3, Word16 var1, Word16 var2)	Msu with rounding
Word32 L_deposit_h(Word16 var1)	16-bit var1 into MSB part
Word32 L_deposit_l(Word16 var1)	16-bit var1 into LSB part
Word32 L_shr_r(Word32 L_var1, Word16 var2)	Long shift right with round
Word32 L_abs(Word32 L_var1)	Long absolute value
Word16 norm_s(Word16 var1)	Short norm
Word16 div_s(Word16 var1, Word16 var2)	Short division
Word16 norm_l(Word32 L_var1)	Long norm

TABLE 12/G.729

Summary of tables fond in tab. ld8.c

Table name	Size	Description
tab_hup_s	28	Upsampling filter for postfilter
tab_hup_1	112	Upsampling filter for postfilter
inter_3	13	FIR filter for interpolating the correlation
inter_3	31	FIR filter for interpolating past excitation
lspcb1	128 × 10	LSP quantizer (first stage)
lspcb2	32 × 10	LSP quantizer (second stage)
fg	2 × 4 × 10	MA predictors in LSP VQ
fg_sum	2 × 10	Used in LSP VQ
fg_sum_inv	2 × 10	Used in LSP VQ
gbk1	8 × 2	Codebook GA in gain VQ
gbk2	16 × 2	Codebook GB in gain VQ
map1	8	Used in gain VQ
imap1	8	Used in gain VQ
map2	16	Used in gain VQ
ima21	16	Used in gain VQ
window	240	LP analysis window
lag_h	10	Lag window for bandwidth expansion (high part)
lag_1	10	Lag window for bandwidth expansion (low part)
grid	61	Grid points in LP to LSP conversion
tabsqr	49	Lookup table in inverse square root computation
tablog	33	Lookup table in base 2 logarithm computation
table	65	Lookup table in LSF to LSP conversion and vice versa
slope	64	Line slopes in LSP to LSF conversion
tabpow	33	Lookup table in 2^x computation

TABLE 13/G.729

Summary of encoder specific routines

Filename	Description
acelp_co.c	Search fixed codebook
cod_1d8k.c	Encoder routine
lpc.c	LP analysis
pitch.c	Pitch search
pre_proc.c	Pre-processing (HP filtering and scaling)
pwf.c	Computation of perceptual weighting coefficients
qua_gain.c	Gain quantizer
qua_1sp.c	LSP quantizer

TABLE 14/G.729

Summary of decoder specific routines

Filename	Description
de_acelp.c	Decode algebraic codebook
dec_gain.c	Decode gains
dec_lag3.c	Decode adaptive-codebook index
dec_ld8k.c	Decoder routine
lspdec.c	LSP decoding routing
post_pro.c	Post processing (HP filtering and scaling)
pst.c	Postfilter routines

TABLE 15/G.729

Summary of general routines

Filename	Description
basicop2.c	Basic operators
oper_32b.c	Extended basic operators
bits.c	Bit manipulation routines
dspfunc.c	Mathematical functions
filter.c	Filter functions
gainpred.c	Gain predictor
lpcfunc.c	Miscellaneous routines related to LP filter
lspgetq.c	LSP quantizer
p_parity.c	Compute pitch parity
pred_lt3.c	Generation of adaptive codebook
util.c	Utility functions